A COMPARE-AGGREGATE MODEL FOR MATCHING TEXT SEQUENCES

Shuohang Wang
School of Information Systems
Singapore Management University
shwang.2014@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

ABSTRACT

Many NLP tasks including machine comprehension, answer selection and text entailment require the comparison between sequences. Matching the important units between sequences is a key to solve these problems. In this paper, we present a general "compare-aggregate" framework that performs word-level matching followed by aggregation using Convolutional Neural Networks. We particularly focus on the different comparison functions we can use to match two vectors. We use four different datasets to evaluate the model. We find that some simple comparison functions based on element-wise operations can work better than standard neural network and neural tensor network.

1 INTRODUCTION

Many natural language processing problems involve matching two or more sequences to make a decision. For example, in textual entailment, one needs to determine whether a hypothesis sentence can be inferred from a premise sentence (Bowman et al., 2015). In machine comprehension, given a passage, a question needs to be matched against it in order to find the correct answer (Richardson et al., 2013; Tapaswi et al., 2016). Table 1 gives two example sequence matching problems. In the first example, a passage, a question and four candidate answers are given. We can see that to get the correct answer, we need to match the question against the passage and identify the last sentence to be the answer-bearing sentence. In the second example, given a question and a set of candidate answers, we need to find the answer that best matches the question. Because of the fundamental importance of comparing two sequences of text to judge their semantic similarity or relatedness, sequence matching has been well studied in natural language processing.

With recent advances of neural network models in natural language processing, a standard practice for sequence modeling now is to encode a sequence of text as an embedding vector using models such as RNN and CNN. To match two sequences, a straightforward approach is to encode each sequence as a vector and then to combine the two vectors to make a decision (Bowman et al., 2015; Feng et al., 2015). However, it has been found that using a single vector to encode an entire sequence is not sufficient to capture all the important information from the sequence, and therefore advanced techniques such as attention mechanisms and memory networks have been applied to sequence matching problems (Hermann et al., 2015; Hill et al., 2016; Rocktäschel et al., 2015).

A common trait of a number of these recent studies on sequence matching problems is the use of a "compare-aggregate" framework (Wang & Jiang, 2016b; He & Lin, 2016; Parikh et al., 2016). In such a framework, comparison of two sequences is not done by comparing two vectors each representing an entire sequence. Instead, these models first compare vector representations of smaller units such as words from these sequences and then aggregate these comparison results to make the final decision. For example, the match-LSTM model proposed by Wang & Jiang (2016b) for textual entailment first compares each word in the hypothesis with an attention-weighted version of the premise. The comparison results are then aggregated through an LSTM. He & Lin (2016) proposed a pairwise word interaction model that first takes each pair of words from two sequences and applies a comparison unit on the two words. It then combines the results of these word interactions using a similarity focus layer followed by a multi-layer CNN. Parikh et al. (2016) proposed a decomposable attention model for textual entailment, in which words from each sequence are compared with an

Plot: ... Aragorn is crowned King of Gondor and taking Arwen as his queen before all present at his coronation bowing before Frodo and the other Hobbits . The Hobbits return to **the Shire** where Sam marries Rosie Cotton	**Question**: can i have auto insurance without a car
Qustion: Where does Sam marry Rosie?	**Ground-truth answer**: yes, it be possible have auto insurance without own a vehicle. you will purchase what be call a name ...
Candidate answers: 0) Grey Havens. 1) Gondor. **2) The Shire**. 3) Erebor. 4) Mordor.	**Another candidate answer**: insurance not be a tax or merely a legal obligation because auto insurance follow a car...

Table 1: The example on the left is a machine comprehension problem from MovieQA, where the correct answer here is **The Shire**. The example on the right is an answer selection problem from InsuranceQA.

attention-weighted version of the other sequence to produce a series of comparison vectors. The comparison vectors are then aggregated and fed into a feed forward network for final classification.

Although these studies have shown the effectiveness of such a "compare-aggregate" framework for sequence matching, there are at least two limitations with these previous studies: (1) Each of the models proposed in these studies is tested on one or two tasks only, but we hypothesize that this general framework is effective on many sequence matching problems. There has not been any study that empirically verifies this. (2) More importantly, these studies did not pay much attention to the comparison function that is used to compare two small textual units. Usually a standard feedforward network is used (Hu et al., 2014; Wang & Jiang, 2016b) to combine two vectors representing two units that need to be compared, e.g., two words. However, based on the nature of these sequence matching problems, we essentially need to measure how semantically similar the two sequences are. Presumably, this property of these sequence matching problems should guide us in choosing more appropriate comparison functions. Indeed He & Lin (2016) used cosine similarity, Euclidean distance and dot product to define the comparison function, which seem to be better justifiable. But they did not systematically evaluate these similarity or distance functions or compare them with a standard feedforward network.

In this paper, we argue that the general "compare-aggregate" framework is effective for a wide range of sequence matching problems. We present a model that follows this general framework and test it on four different datasets, namely, MovieQA, InsuranceQA, WikiQA and SNLI. The first three datasets are for Question Answering, but the setups of the tasks are quite different. The last dataset is for textual entailment. More importantly, we systematically present and test six different comparison functions. We find that overall a comparison function based on element-wise subtraction and multiplication works the best on the four datasets.

The contributions of this work are twofold: (1) Using four different datasets, we show that our model following the "compare-aggregate" framework is very effective when compared with the state-of-the-art performance on these datasets. (2) We conduct systematic evaluation of different comparison functions and show that a comparison function based on element-wise operations, which is not widely used for word-level matching, works the best across the different datasets. We believe that these findings will be useful for future research on sequence matching problems. We have also made our code available online.[1]

2 METHOD

In this section, we propose a general model following the "compare-aggregate" framework for matching two sequences. This general model can be applied to different tasks. We focus our discussion on six different comparison functions that can be plugged into this general "compare-aggregate" model. In particular, we hypothesize that two comparison functions based on element-wise operations, SUB and MULT, are good middle ground between highly flexible functions using standard neural network models and highly restrictive functions based on cosine similarity and/or Euclidean

[1] https://github.com/shuohangwang/SeqMatchSeq

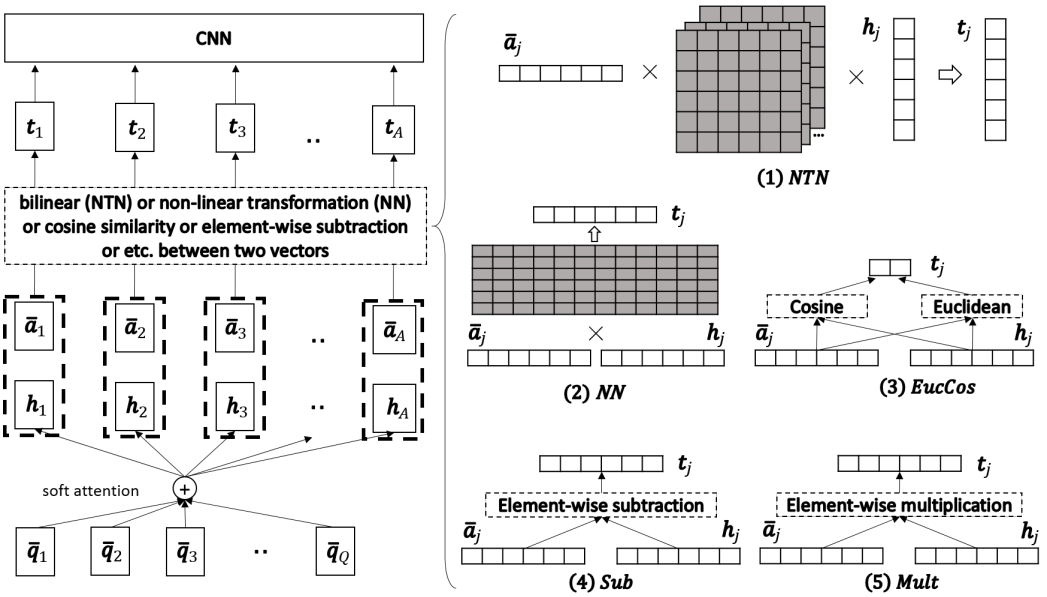

Figure 1: The left hand side is an overview of the model. The right hand side shows the details about the different comparison functions. The rectangles in dark represent parameters to be learned. $\times$ represents matrix multiplication.

distance. As we will show in the experiment section, these comparison functions based on element-wise operations can indeed perform very well on a number of sequence matching problems.

2.1 Problem Definition and Model Overview

The general setup of the sequence matching problem we consider is the following. We assume there are two sequences to be matched. We use two matrices $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ and $\mathbf{A} \in \mathbb{R}^{d \times A}$ to represent the word embeddings of the two sequences, where Q and A are the lengths of the two sequences, respectively, and d is the dimensionality of the word embeddings. In other words, each column vector of $\mathbf{Q}$ or $\mathbf{A}$ is an embedding vector representing a single word. Given a pair of $\mathbf{Q}$ and $\mathbf{A}$, the goal is to predict a label y. For example, in textual entailment, $\mathbf{Q}$ may represent a premise and $\mathbf{A}$ a hypothesis, and y indicates whether $\mathbf{Q}$ entails $\mathbf{A}$ or contradicts $\mathbf{A}$. In question answering, $\mathbf{Q}$ may be a question and $\mathbf{A}$ a candidate answer, and y indicates whether $\mathbf{A}$ is the correct answer to $\mathbf{Q}$.

We treat the problem as a supervised learning task. We assume that a set of training examples in the form of $(\mathbf{Q}, \mathbf{A}, y)$ is given and we aim to learn a model that maps any pair of $(\mathbf{Q}, \mathbf{A})$ to a y.

An overview of our model is shown in Figure 1. The model can be divided into the following four layers:

1. **Preprocessing:** We use a preprocessing layer (not shown in the figure) to process $\mathbf{Q}$ and $\mathbf{A}$ to obtain two new matrices $\overline{\mathbf{Q}} \in \mathbb{R}^{l \times Q}$ and $\overline{\mathbf{A}} \in \mathbb{R}^{l \times A}$. The purpose here is to use some gate values to control the importance of different words in making the predictions on the sequence pair. For example, $\overline{\mathbf{q}}_i \in \mathbb{R}^l$, which is the i^{th} column vector of $\overline{\mathbf{Q}}$, encodes the i^{th} word in $\mathbf{Q}$.
2. **Attention:** We apply a standard attention mechanism on $\overline{\mathbf{Q}}$ and $\overline{\mathbf{A}}$ to obtain attention weights over the column vectors in $\overline{\mathbf{Q}}$ for each column vector in $\overline{\mathbf{A}}$. With these attention weights, for each column vector $\overline{\mathbf{a}}_j$ in $\overline{\mathbf{A}}$, we obtain a corresponding vector $\mathbf{h}_j$, which is an attention-weighted sum of the column vectors of $\overline{\mathbf{Q}}$.
3. **Comparison:** We use a comparison function f to combine each pair of $\overline{\mathbf{a}}_j$ and $\mathbf{h}_j$ into a vector $\mathbf{t}_j$.

4. **Aggregation:** We use a CNN layer to aggregate the sequence of vectors $\mathbf{t}_j$ for the final classification.

Although this model follows more or less the same framework as the model proposed by Parikh et al. (2016), our work has some notable differences. First, we will pay much attention to the comparison function f and compare a number of options, including some uncommon ones based on element-wise operations. Second, we apply our model to four different datasets representing four different tasks to evaluate its general effectiveness for sequence matching problems. There are also some other differences from the work by Parikh et al. (2016). For example, we use a CNN layer instead of summation and concatenation for aggregation. Our attention mechanism is one-directional instead of two-directional.

In the rest of this section we will present the model in detail. We will focus mostly on the comparison functions we consider.

2.2 Preprocessing and Attention

Inspired by the use of gates in LSTM and GRU, we preprocess $\mathbf{Q}$ and $\mathbf{A}$ with the following formulas:

$$\begin{aligned} \overline{\mathbf{Q}} &= \sigma(\mathbf{W}^{\mathrm{i}}\mathbf{Q} + \mathbf{b}^{\mathrm{i}} \otimes \mathbf{e}_Q) \odot \tanh(\mathbf{W}^{\mathrm{u}}\mathbf{Q} + \mathbf{b}^{\mathrm{u}} \otimes \mathbf{e}_Q), \\ \overline{\mathbf{A}} &= \sigma(\mathbf{W}^{\mathrm{i}}\mathbf{A} + \mathbf{b}^{\mathrm{i}} \otimes \mathbf{e}_A) \odot \tanh(\mathbf{W}^{\mathrm{u}}\mathbf{A} + \mathbf{b}^{\mathrm{u}} \otimes \mathbf{e}_A), \end{aligned} \tag{1}$$

where $\odot$ is element-wise multiplication, and $\mathbf{W}^{\mathrm{i}}, \mathbf{W}^{\mathrm{u}} \in \mathbb{R}^{l \times d}$ and $\mathbf{b}^{\mathrm{i}}, \mathbf{b}^{\mathrm{u}} \in \mathbb{R}^{l}$ are parameters to be learned. The outer product $(\cdot \otimes \mathbf{e}_X)$ produces a matrix or row vector by repeating the vector or scalar on the left for X times. Here $\sigma(\mathbf{W}^{\mathrm{i}}\mathbf{Q} + \mathbf{b}^{\mathrm{i}} \otimes \mathbf{e}_Q)$ and $\sigma(\mathbf{W}^{\mathrm{i}}\mathbf{A} + \mathbf{b}^{\mathrm{i}} \otimes \mathbf{e}_A)$ act as gate values to control the degree to which the original values of $\mathbf{Q}$ and $\mathbf{A}$ are preserved in $\overline{\mathbf{Q}}$ and $\overline{\mathbf{A}}$. For example, for stop words, their gate values would likely be low for tasks where stop words make little difference to the final predictions.

In this preprocessing step, the word order does not matter. Although a better way would be to use RNN such as LSTM and GRU to chain up the words such that we can capture some contextual information, this could be computationally expensive for long sequences. In our experiments, we only incorporated LSTM into the formulas above for the SNLI task.

The general attention (Luong et al., 2015) layer is built on top of the resulting $\overline{\mathbf{Q}}$ and $\overline{\mathbf{A}}$ as follows:

$$\begin{aligned} \mathbf{G} &= \mathrm{softmax}\left((\mathbf{W}^{\mathrm{g}}\overline{\mathbf{Q}} + \mathbf{b}^{\mathrm{g}} \otimes \mathbf{e}_Q)^{\mathrm{T}}\overline{\mathbf{A}}\right), \\ \mathbf{H} &= \overline{\mathbf{Q}}\mathbf{G}, \end{aligned} \tag{2}$$

where $\mathbf{W}^{\mathrm{g}} \in \mathbb{R}^{l \times l}$ and $\mathbf{b}^{\mathrm{g}} \in \mathbb{R}^{l}$ are parameters to be learned, $\mathbf{G} \in \mathbb{R}^{Q \times A}$ is the attention weight matrix, and $\mathbf{H} \in \mathbb{R}^{l \times A}$ are the attention-weighted vectors. Specifically, $\mathbf{h}_j$, which is the j^{th} column vector of $\mathbf{H}$, is a weighted sum of the column vectors of $\overline{\mathbf{Q}}$ and represents the part of $\mathbf{Q}$ that best matches the j^{th} word in $\mathbf{A}$. Next we will combine $\mathbf{h}_j$ and $\overline{\mathbf{a}}_j$ using a comparison function.

2.3 Comparison

The goal of the comparison layer is to match each $\overline{\mathbf{a}}_j$, which represents the j^{th} word and its context in $\mathbf{A}$, with $\mathbf{h}_j$, which represents a weighted version of $\mathbf{Q}$ that best matches $\overline{\mathbf{a}}_j$. Let f denote a comparison function that transforms $\overline{\mathbf{a}}_j$ and $\mathbf{h}_j$ into a vector $\mathbf{t}_j$ to represent the comparison result.

A natural choice of f is a standard neural network layer that consists of a linear transformation followed by a non-linear activation function. For example, we can consider the following choice:

$$\text{NeuralNet (NN):} \qquad \mathbf{t}_j = f(\overline{\mathbf{a}}_j, \mathbf{h}_j) = \mathrm{ReLU}(\mathbf{W}\begin{bmatrix}\overline{\mathbf{a}}_j \\ \mathbf{h}_j\end{bmatrix} + \mathbf{b}), \tag{3}$$

where matrix $\mathbf{W} \in \mathbb{R}^{l \times 2l}$ and vector $\mathbf{b} \in \mathbb{R}^{l}$ are parameters to be learned.

Alternatively, another natural choice is a neural tensor network (Socher et al., 2013) as follows:

$$\text{NeuralTensorNet (NTN):} \qquad \mathbf{t}_j = f(\overline{\mathbf{a}}_j, \mathbf{h}_j) = \mathrm{ReLU}(\overline{\mathbf{a}}_j^{\mathrm{T}}\mathbf{T}^{[1\ldots l]}\mathbf{h}_j + \mathbf{b}), \tag{4}$$

where tensor $\mathbf{T}^{[1\ldots l]} \in \mathbb{R}^{l \times l \times l}$ and vector $\mathbf{b} \in \mathbb{R}^{l}$ are parameters to be learned.

However, we note that for many sequence matching problems, we intend to measure the semantic similarity or relatedness of the two sequences. So at the word level, we also intend to check how similar or related $\bar{\mathbf{a}}_j$ is to $\mathbf{h}_j$. For this reason, a more natural choice used in some previous work is Euclidean distance or cosine similarity between $\bar{\mathbf{a}}_j$ and $\mathbf{h}_j$. We therefore consider the following definition of f:

$$\text{EUCLIDEAN+COSINE (EUCCOS):} \qquad \mathbf{t}_j = f(\bar{\mathbf{a}}_j, \mathbf{h}_j) = \begin{bmatrix} \|\bar{\mathbf{a}}_j - \mathbf{h}_j\|_2 \\ \cos(\bar{\mathbf{a}}_j, \mathbf{h}_j) \end{bmatrix}. \tag{5}$$

Note that with EUCCOS, the resulting vector $\mathbf{t}_j$ is only a 2-dimensional vector. Although EUCCOS is a well-justified comparison function, we suspect that it may lose some useful information from the original vectors $\bar{\mathbf{a}}_j$ and $\mathbf{h}_j$. On the other hand, NN and NTN are too general and thus do not capture the intuition that we care mostly about the similarity between $\bar{\mathbf{a}}_j$ and $\mathbf{h}_j$.

To use something that is a good compromise between the two extreme cases, we consider the following two new comparison functions, which operate on the two vectors in an element-wise manner. These functions have been used previously by Mou et al. (2016).

$$\text{SUBTRACTION (SUB):} \qquad \mathbf{t}_j = f(\bar{\mathbf{a}}_j, \mathbf{h}_j) = (\bar{\mathbf{a}}_j - \mathbf{h}_j) \odot (\bar{\mathbf{a}}_j - \mathbf{h}_j), \tag{6}$$

$$\text{MULTIPLICATION (MULT):} \qquad \mathbf{t}_j = f(\bar{\mathbf{a}}_j, \mathbf{h}_j) = \bar{\mathbf{a}}_j \odot \mathbf{h}_j. \tag{7}$$

Note that the operator $\odot$ is element-wise multiplication. For both comparison functions, the resulting vector $\mathbf{t}_j$ has the same dimensionality as $\bar{\mathbf{a}}_j$ and $\mathbf{h}_j$.

We can see that SUB is closely related to Euclidean distance in that Euclidean distance is the sum of all the entries of the vector $\mathbf{t}_j$ produced by SUB. But by not summing up these entries, SUB preserves some information about the different dimensions of the original two vectors. Similarly, MULT is closely related to cosine similarity but preserves some information about the original two vectors.

Finally, we consider combining SUB and MULT followed by an NN layer as follows:

$$\text{SUBMULT+NN:} \qquad \mathbf{t}_j = f(\bar{\mathbf{a}}_j, \mathbf{h}_j) = \text{ReLU}(\mathbf{W} \begin{bmatrix} (\bar{\mathbf{a}}_j - \mathbf{h}_j) \odot (\bar{\mathbf{a}}_j - \mathbf{h}_j) \\ \bar{\mathbf{a}}_j \odot \mathbf{h}_j \end{bmatrix} + \mathbf{b}). \tag{8}$$

In summary, we consider six different comparison functions: NN, NTN, EUCCOS, SUB, MULT and SUBMULT+NN. Among these functions, the last three (SUB, MULT and SUBMULT+NN) have not been widely used in previous work for word-level matching.

2.4 AGGREGATION

After we apply the comparison function to each pair of $\bar{\mathbf{a}}_j$ and $\mathbf{h}_j$ to obtain a series of vectors $\mathbf{t}_j$, finally we aggregate these vectors using a one-layer CNN (Kim, 2014):

$$\mathbf{r} \quad = \quad \text{CNN}([\mathbf{t}_1, \ldots, \mathbf{t}_A]). \tag{9}$$

$\mathbf{r} \in \mathbb{R}^{nl}$ is then used for the final classification, where n is the number of windows in CNN.

3 EXPERIMENTS

	MovieQA			InsuranceQA			WikiQA			SNLI		
	train	dev	test	train	dev	test	train	dev	test	train	dev	test
#Q	9848	1958	3138	13K	1K	1.8K*2	873	126	243	549K	9842	9824
#C	5	5	5	50	500	500	10	9	10	-	-	-
#w in P	873	866	914	-	-	-	-	-	-	-	-	-
#w in Q	10.6	10.6	10.8	7.2	7.2	7.2	6.5	6.5	6.4	14	15.2	15.2
#w in A	5.9	5.6	5.5	92.1	92.1	92.1	25.5	24.7	25.1	8.3	8.4	8.3

Table 2: The statistics of different datasets. Q:question/hypothesis, C:candidate answers for each question, A:answer/hypothesis, P:plot, w:word (average).

Models	MovieQA		InsuranceQA			WikiQA		SNLI	
	dev	test	dev	test1	test2	MAP	MRR	train	test
Cosine Word2Vec	46.4	45.63	-	-	-	-	-	-	-
Cosine TFIDF	47.6	**47.36**	-	-	-	-	-	-	-
SSCB TFIDF	**48.5**	-	-	-	-	-	-	-	-
IR model	-	-	52.7	55.1	50.8	-	-	-	-
CNN with GESD	-	-	65.4	65.3	61.0	-	-	-	-
Attentive LSTM	-	-	68.9	69.0	64.8	-	-	-	-
IARNN-Occam	-	-	69.1	68.9	**65.1**	**0.7341**	**0.7418**	-	-
IARNN-Gate	-	-	**70.0**	**70.1**	62.8	0.7258	0.7394	-	-
CNN-Cnt	-	-	-	-	-	0.6520	0.6652	-	-
ABCNN	-	-	-	-	-	0.6921	0.7108	-	-
CubeCNN	-	-	-	-	-	0.7090	0.7234	-	-
W-by-W Attention	-	-	-	-	-	-	-	85.3	83.5
match-LSTM	-	-	-	-	-	-	-	92.0	86.1
LSTMN	-	-	-	-	-	-	-	88.5	86.3
Decomp Attention	-	-	-	-	-	-	-	90.5	86.8
EBIM+TreeLSTM	-	-	-	-	-	-	-	93.0	**88.3**
NN	31.6	-	76.8	74.9	72.4	0.7102	0.7224	89.3	86.3
NTN	31.6	-	75.6	75.0	72.5	0.7349	0.7456	91.6	86.3
EUCCOS	71.9	-	70.6	70.2	67.9	0.6740	0.6882	87.1	84.0
SUB	64.9	-	70.0	71.3	68.2	0.7019	0.7151	89.8	**86.8**
MULT	66.4	-	76.0	75.2	**73.4**	**0.7433**	**0.7545**	89.7	85.8
SUBMULT+NN	**72.1**	**72.9**	**77.0**	**75.6**	72.3	0.7332	0.7477	89.4	**86.8**

Table 3: Experiment Results

Models	MovieQA		InsuranceQA			WikiQA		SNLI	
	dev	test	dev	test1	test2	MAP	MRR	train	test
SUBMULT+NN (no preprocess)	72.0	-	72.8	73.8	70.7	0.6996	0.7156	89.6	82.8
SUBMULT+NN (no attention)	60.4	-	69.4	70.4	67.8	0.7164	0.7238	89.0	84.4

Table 4: Ablation Experiment Results. "no preprocess": remove the preprocessing layer by directly using word embeddings $\mathbf{Q}$ and $\mathbf{A}$ to replace $\overline{\mathbf{Q}}$ and $\overline{\mathbf{A}}$ in Eqn. 1; "no attention": remove the attention layer by using mean pooling of $\overline{\mathbf{Q}}$ to replace all the vectors of $\mathbf{H}$ in Eqn. 2.

In this section, we evaluate our model on four different datasets representing different tasks. The first three datasets are question answering tasks while the last one is on textual entailment. The statistics of the four datasets are shown in Table 2. We will fist introduce the task settings and the way we customize the "compare-aggregate" structure to each task. Then we will show the baselines for the different datasets. Finally, we discuss the experiment results shown in Table 3 and the ablation study shown in Table 4.

3.1 TASK-SPECIFIC MODEL STRUCTURES

In all these tasks, we use matrix $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ to represent the question or premise and matrix $\mathbf{A}_k \in \mathbb{R}^{d \times A_k}$ ($k \in [1, K]$) to represent the k^{th} answer or the hypothesis. For the machine comprehension task **MovieQA** (Tapaswi et al., 2016), there is also a matrix $\mathbf{P} \in \mathbb{R}^{d \times P}$ that represents the plot of a movie. Here Q is the length of the question or premise, A_k the length of the k^{th} answer, and P the length of the plot.

For the **SNLI** (Bowman et al., 2015) dataset, the task is text entailment, which identifies the relationship (entailment, contradiction or neutral) between a premise sentence and a hypothesis sentence. Here $K = 1$, and there are exactly two sequences to match. The actual model structure is what we have described before.

For the **InsuranceQA** (Feng et al., 2015) dataset, the task is an answer selection task which needs to select the correct answer for a question from a candidate pool. For the **WikiQA** (Yang et al., 2015) datasets, we need to rank the candidate answers according to a question. For both tasks,

there are K candidate answers for each question. Let us use $\mathbf{r}_k$ to represent the resulting vector produced by Eqn. 9 for the k^{th} answer. In order to select one of the K answers, we first define $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_K]$. We then compute the probability of the k^{th} answer to be the correct one as follows:

$$p(k|\mathbf{R}) \quad = \quad \text{softmax}(\mathbf{w}^{\text{T}} \tanh(\mathbf{W}^{\text{s}}\mathbf{R} + \mathbf{b}^{\text{s}} \otimes \mathbf{e}_K) + b \otimes \mathbf{e}_K), \tag{10}$$

where $\mathbf{W}^{\text{s}} \in \mathbb{R}^{l \times nl}$, $\mathbf{w} \in \mathbb{R}^l$, $\mathbf{b}^{\text{s}} \in \mathbb{R}^l$, $b \in \mathbb{R}$ are parameters to be learned.

For the machine comprehension task **MovieQA**, each question is related to Plot Synopses written by fans after watching the movie and each question has five candidate answers. So for each candidate answer there are three sequences to be matched: the plot $\mathbf{P}$, the question $\mathbf{Q}$ and the answer $\mathbf{A}_k$. For each k, we first match $\mathbf{Q}$ and $\mathbf{P}$ and refer to the matching result at position j as $\mathbf{t}^{\text{q}}_j$, as generated by one of the comparison functions f. Similarly, we also match $\mathbf{A}_k$ with $\mathbf{P}$ and refer to the matching result at position j as $\mathbf{t}^{\text{a}}_{k,j}$. We then define

$$\mathbf{t}_{k,j} \quad = \quad \begin{bmatrix} \mathbf{t}^{\text{q}}_j \\ \mathbf{t}^{\text{a}}_{k,j} \end{bmatrix},$$

and

$$\mathbf{r}_k \quad = \quad \text{CNN}([\mathbf{t}_{k,1}, \ldots, \mathbf{t}_{k,P}]).$$

To select an answer from the K candidate answers, again we use Eqn. 10 to compute the probabilities.

The implementation details of the modes are as follows. The word embeddings are initialized from GloVe (Pennington et al., 2014). During training, they are not updated. The word embeddings not found in GloVe are initialized with zero.

The dimensionality l of the hidden layers is set to be 150. We use ADAMAX (Kingma & Ba, 2015) with the coefficients $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to optimize the model. We do not use L2-regularization. The main parameter we tuned is the dropout on the embedding layer. For WikiQA, which is a relatively small dataset, we also tune the learning rate and the batch size. For the others, we set the batch size to be 30 and the learning rate 0.002.

3.2 BASELINES

Here, we will introduce the baselines for each dataset. We did not re-implement these models but simply took the reported performance for the purpose of comparison.

SNLI: • **W-by-W Attention**: The model by Rocktäschel et al. (2015), who first introduced attention mechanism into text entailment. • **match-LSTM**: The model by Wang & Jiang (2016b), which concatenates the matched words as the inputs of an LSTM. • **LSTMN**: Long short-term memory-networks proposed by Cheng et al. (2016). • **Decomp Attention**: Another "compare-aggregate" model proposed by Parikh et al. (2016). • **EBIM+TreeLSTM**: The state-of-the-art model proposed by Chen et al. (2016) on the SNLI dataset.

InsuranceQA: • **IR model**: This model by Bendersky et al. (2010) learns the concept information to help rank the candidates. • **CNN with GESD**: This model by Feng et al. (2015) uses Euclidean distance and dot product between sequence representations built through convolutional neural networks to select the answer. • **Attentive LSTM**: Tan et al. (2016) used soft-attention mechanism to select the most important information from the candidates according to the representation of the questions. • **IARNN-Occam**: This model by Wang et al. (2016) adds regularization on the attention weights. • **IARNN-Gate**: This model by Wang et al. (2016) uses the representation of the question to build the GRU gates for each candidate answer.

WikiQA: • **IARNN-Occam** and **IARNN-Gate** as introduced before. • **CNN-Cnt**: This model by Yang et al. (2015) combines sentence representations built by a convolutional neural network with logistic regression. • **ABCNN**: This model is Attention-Based Convolutional Neural Network proposed by Yin et al. (2015). • **CubeCNN** proposed by He & Lin (2016) builds a CNN on all pairs of word similarity.

MovieQA: All the baselines we consider come from Tapaswi et al. (2016)'s work: • **Cosine Word2Vec**: A sliding window is used to select the answer according to the similarities computed

through Word2Vec between the sentences in plot and the question/answer. • **Cosine TFIDF**: This model is similar to the previous method but uses bag-of-word with tf-idf scores to compute similarity. • **SSCB TFIDF**: Instead of using the sliding window method, a convolutional neural network is built on the sentence level similarities.

3.3 Analysis of Results

We use accuracy as the evaluation metric for the datasets MovieQA, InsuranceQA and SNLI, as there is only one correct answer or one label for each instance. For WikiQA, there may be multiple correct answers, so evaluation metrics we use are Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

We observe the following from the results. (1) Overall, we can find that our general "compare-aggregate" structure achieves the best performance on **MovieQA**, **InsuranceQA**, **WikiQA** datasets and very competitive performance on the **SNLI** dataset. Especially for the **InsuranceQA** dataset, with any comparison function we use, our model can outperform all the previous models. (2) The comparison method SubMult+NN is the best in general. (3) Some simple comparison functions can achieve better performance than the neural networks or neural tensor network comparison functions. For example, the simplest comparison function EucCos achieves nearly the best performance in the **MovieQA** dataset, and the element-wise comparison functions, which do not need parameters can achieve the best performance on the **WikiQA** dataset. (4) We find the preprocessing layer and the attention layer for word selection to be important in the "compare-aggregate" structure through the experiments of removing these two layers separately. We also see that for sequence matching with big difference in length, such as the **MovieQA** and **InsuranceQA** tasks, the attention layer plays a more important role. For sequence matching with smaller difference in length, such as the **WikiQA** and **SNLI** tasks, the pre-processing layer plays a more important role. (5) For the **MovieQA**, **InsuranceQA** and **WikiQA** tasks, our preprocessing layer is order-insensitive so that it will not take the context information into consideration during the comparison, but our model can still outperform the previous work with order-sensitive preprocessing layer. With this finding, we believe the word-by-word comparison part plays a very important role in these tasks. We will further explore the preprocessing layer in the future.

3.4 Further Analyses

To further explain how our model works, we visualize the max values in each dimension of the convolutional layer. We use two examples shown in Table 1 from MovieQA and InsuranceQA datasets respectively. In the top of Figure 2, we can see that the plot words that also appear in either the question or the answer will draw more attention by the CNN. We hypothesize that if the nearby words in the plot can match both the words in question and the words in one answer, then this answer is more likely to be the correct one. Similarly, the bottom one of Figure 2 also shows that the CNN will focus more on the matched word representations. If the words in one answer continuously match the words in the question, this answer is more likely to be the correct one.

4 Related Work

We review related work in three types of general structures for matching sequences.

Siamense network: These kinds of models use the same structure, such as RNN or CNN, to build the representations for the sequences separately and then use them for classification. Then cosine similarity (Feng et al., 2015; Yang et al., 2015), element-wise operation (Tai et al., 2015; Mou et al., 2016) or neural network-based combination Bowman et al. (2015) are used for sequence matching.

Attentive network: Soft-attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) has been widely used for sequence matching in machine comprehension (Hermann et al., 2015), text entailment (Rocktäschel et al., 2015) and question answering (Tan et al., 2016). Instead of using the final state of RNN to represent a sequence, these studies use weighted sum of all the states for the sequence representation.

Compare-Aggregate network: This kind of framework is to perform the word level matching (Wang & Jiang, 2016a; Parikh et al., 2016; He & Lin, 2016; Trischler et al., 2016; Wan et al.,

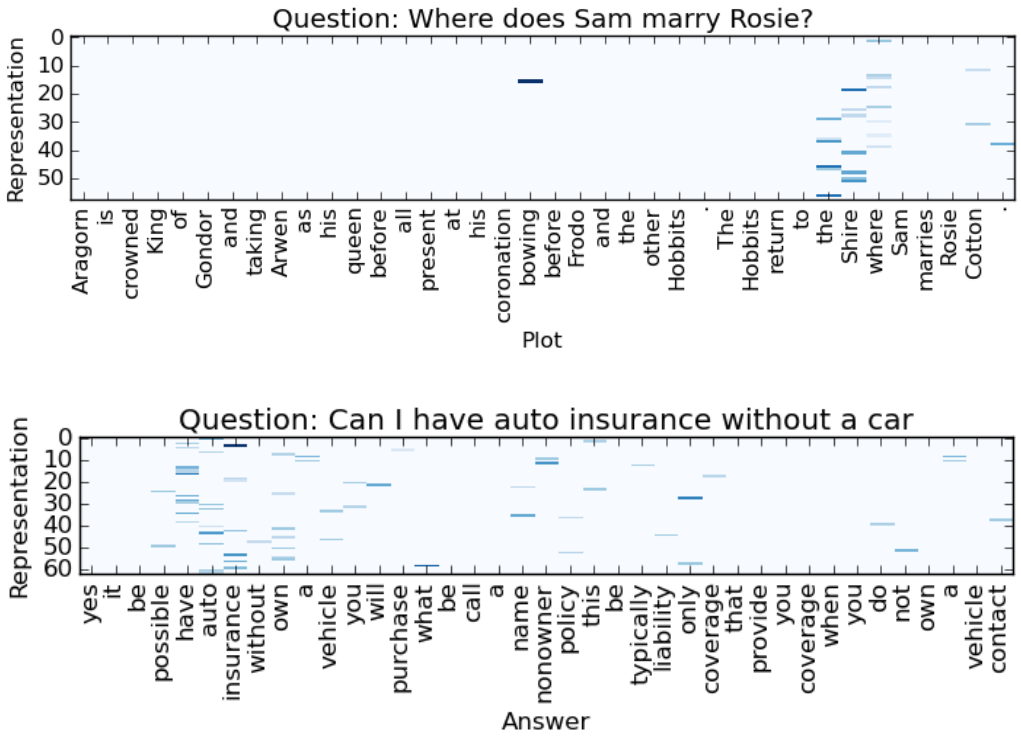

Figure 2: An visualization of the largest value of each dimension in the convolutional layer of CNN. The top figure is an example from the dataset **MovieQA** with CNN window size 5. The bottom figure is an example from the dataset **InsuranceQA** with CNN window size 3. Due to the sparsity of the representation, we show only the dimensions with larger values. The dimensionality of the raw representations is 150.

2016). Our work is under this framework. But our structure is different from previous models and our model can be applied on different tasks. Besides, we analyzed different word-level comparison functions separately.

5 Conclusions

In this paper, we systematically analyzed the effectiveness of a "compare-aggregate" model on four different datasets representing different tasks. Moreover, we compared and tested different kinds of word-level comparison functions and found that some element-wise comparison functions can outperform the others. According to our experiment results, many different tasks can share the same "compare-aggregate" structure. In the future work, we would like to test its effectiveness on multi-task learning.

6 Acknowledgments

This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2014.

Michael Bendersky, Donald Metzler, and W Bruce Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*. ACM, 2010.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. Enhancing and combining sequential and tree LSTM for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 813–820. IEEE, 2015.

Hua He and Jimmy Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2015.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the International Conference on Learning Representations*, 2016.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, 2014.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. In *Proceedings of the International Conference on Learning Representations*, 2015.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the Conference on Association for Computational Linguistics*, 2015.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Improved representation learning for question answer matching. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. MovieQA: Understanding stories in movies through question-answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. A parallel-hierarchical model for machine comprehension on sparse data. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. Match-srnn: Modeling the recursive matching structure with spatial RNN. *International Joint Conference on Artificial Intelligence*, 2016.

Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Shuohang Wang and Jing Jiang. Machine comprehension using match-LSTM and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016a.

Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. In *Proceedings of the Conference on the North American Chapter of the Association for Computational Linguistics*, 2016b.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015.